武汉纺织大学

Java应用开发课程设计

**基于MyBatis的图书馆系统**

**学 院： 数学与计算机学院**

**班 级： 物联网11802**

**姓 名： 洪昊明**

**学 号： 1814290133**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年6月20日**

目 录

[1 需求分析 1](#_Toc42813635)

[1.1用户登陆 1](#_Toc42813636)

[1.2菜单 1](#_Toc42813637)

[1.3各项功能 2](#_Toc42813638)

[1.3.1 读者借阅系统功能 2](#_Toc42813639)

[1.3.2 图书管理员管理系统 3](#_Toc42813640)

[1.3.3 退出 4](#_Toc42813642)

[2 系统设计 4](#_Toc42813643)

[2.1用户用例图 4](#_Toc42813644)

[2.2 UML类图（Class Diagram） 6](#_Toc42813646)

[2.3 UML时序图（Sequence Diagram） 8](#_Toc42813647)

[2.3.1 系统登录 8](#_Toc42813648)

[2.4 UML活动图（Activity Diagram） 8](#_Toc42813651)

[2.4.1 登陆 8](#_Toc42813652)

[2.4.1 借阅 9](#_Toc42813653)

[3 系统实现 11](#_Toc42813656)

[3.1 项目结构 11](#_Toc42813657)

[3.2 配置文件 11](#_Toc42813658)

[3.2.1 jdbc.properties文件 11](#_Toc42813659)

[3.2.2 log4j.properties文件 11](#_Toc42813660)

[3.2.3 mybatis-config.xml文件 12](#_Toc42813661)

[3.3 VO类User.java 12](#_Toc42813662)

[3.4 DAO接口类IUserDAO.java 13](#_Toc42813663)

[3.5 接口映射文件UserMapper.xml 13](#_Toc42813664)

[3.6 工具包Util 13](#_Toc42813665)

[3.6.1 MybatisUtils.java 13](#_Toc42813666)

[3.6.2 MD5Util.java 14](#_Toc42813667)

[3.7 服务层UserService.java 15](#_Toc42813668)

[3.8 用户界面Driver.java 16](#_Toc42813669)

[3.9 \*\*\*\*\*\*.java 16](#_Toc42813670)

[4 系统测试 16](#_Toc42813671)

[5 系统总结 16](#_Toc42813672)

# 0 前言

本次试验较为综合，不得不说这是这几个期末选题中最棘手的三个之一，更何况诚实的说本课程后半段听得不多，最后几个作业硬是憋不出来，本次作业也是在同（和我选题不一样）学的帮助和教导之下，在对同学的最后一次作业的参考还有一位好心学长（然而他不熟悉eclipse）手把手教的情况下勉强完成，实属不易。

本次疫情确实给同学们的上课和老师的教学带来了诸多不便，确实是辛苦老师了，我在最后一节课发表感言的时候说到过，我以后也并没有要从业计算机行业的打算，这次的假期给我带来了非常宝贵的学习时间（学习电影），但同时也一定程度上让我减弱了我对本科的学业的重视程度，我认为一切都是选择，也无所谓后悔不后悔，事在人为，做就好了，当然我还是希望老师能放我过的，毕竟大部分时间我还是有听课的（虽然不一定听懂），望老师“手抖一些”。

在此也祝愿武汉情况越来越好，早日开学吧。

# 1 需求分析

采用MyBatis框架实现基于DAO设计模式的图书馆系统：

用户角色划分：读者、图书馆人员

各角色基本功能如下：

读者角色：登陆，查询图书，借阅图书，归还图书，查询借阅记录，修改密码，

图书馆人员角色：登陆，图书增加，图书修改，图书删除，读者增加，读者修改，读者删除，借阅查询，借阅统计等。

自行完成需求分析，参照文档模板写出系统详细的需求分析。

绘制UML的用户用例图（Use Case）

设计对应的数据库，绘制数据库的ER图

完成每个类的设计，绘制系统的类图

对于每一个业务逻辑，绘制UML时序图

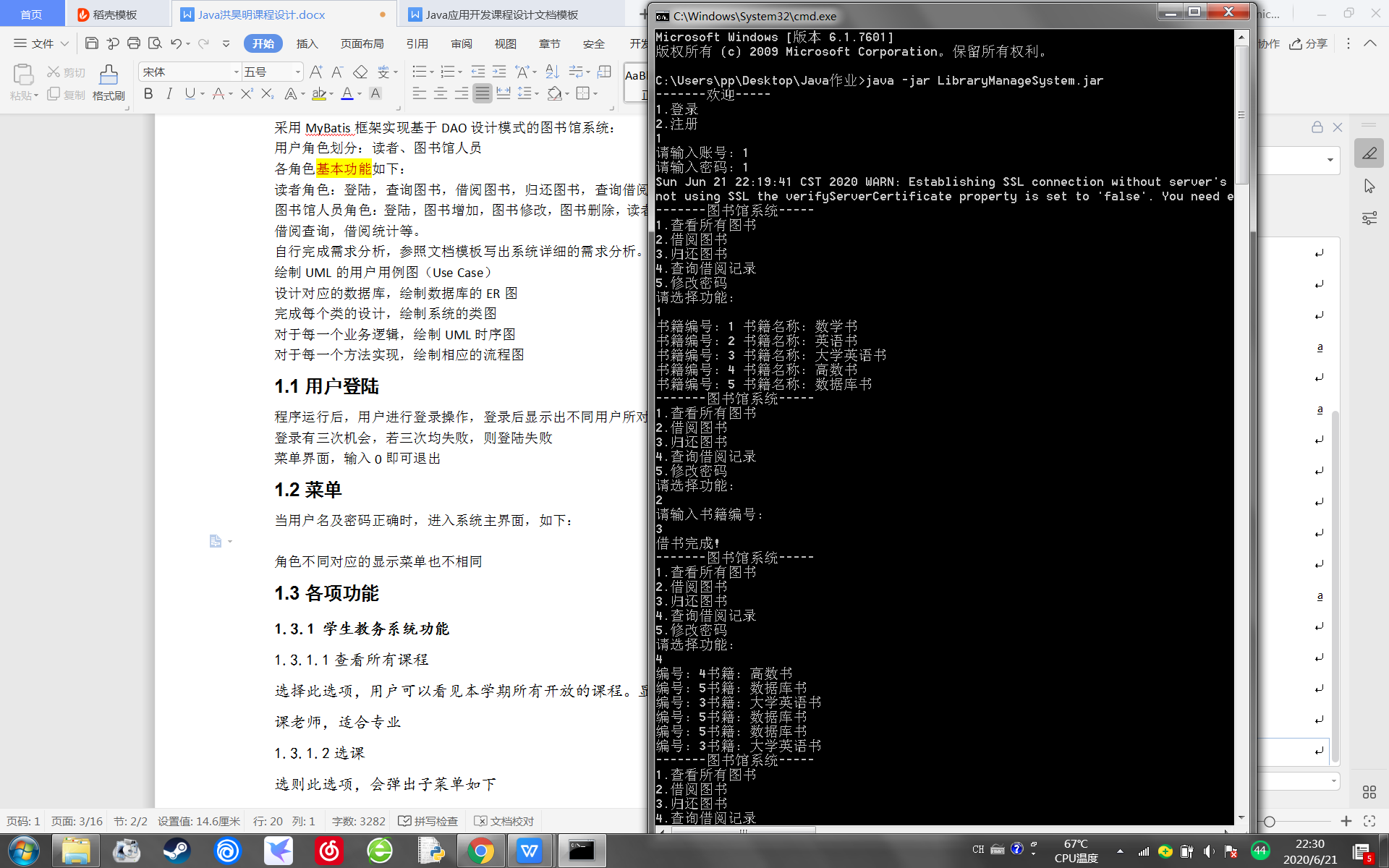
对于每一个方法实现，绘制相应的流程图

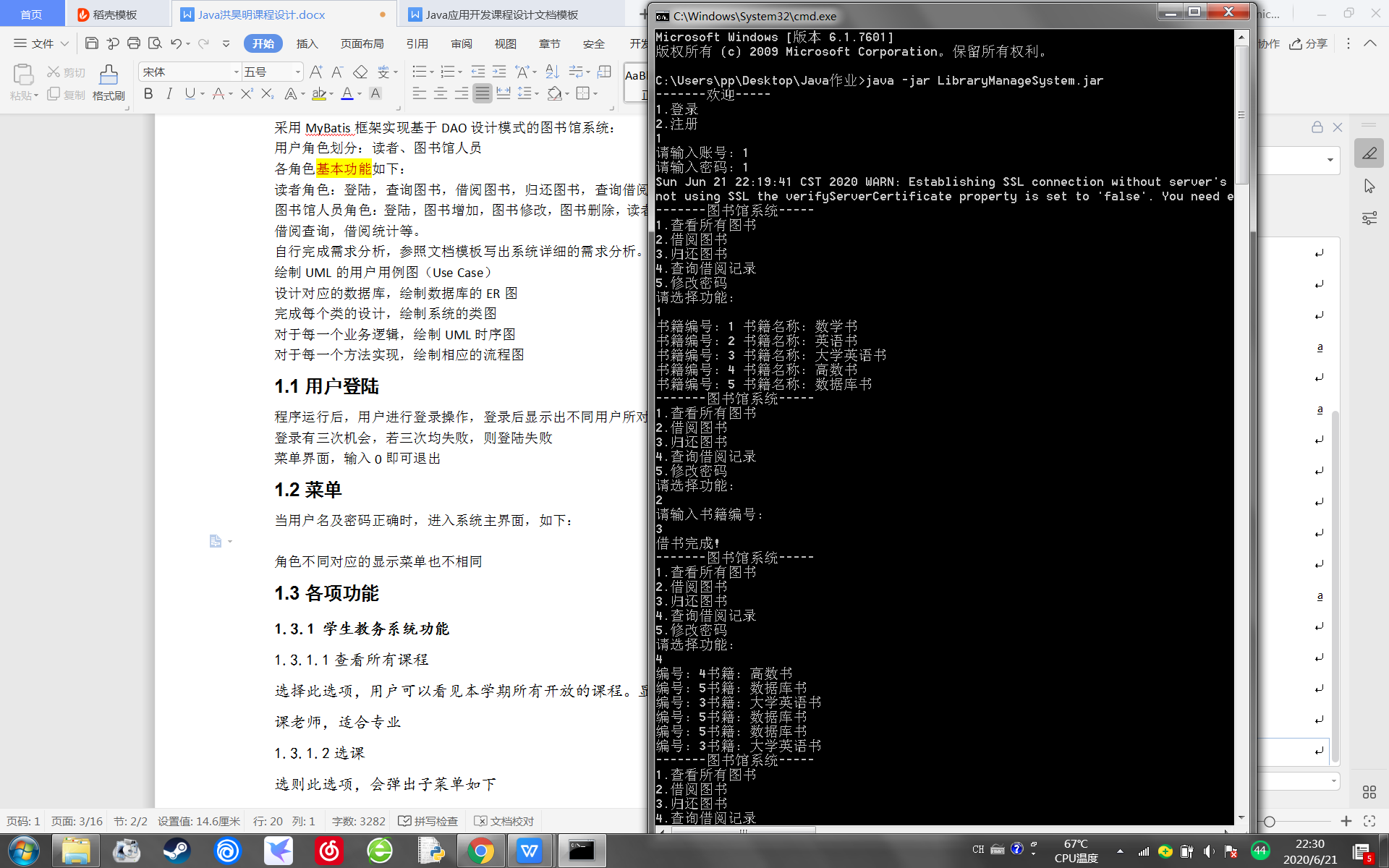
## 1.1用户登陆

程序运行后，用户进行登录操作，（账号1密码1为读者，全为2则为图书管理员），菜单界面，输入0即可退出

## 1.2菜单

当用户名及密码正确时，进入系统主界面，如下：





角色不同对应的显示菜单也不相同

## 1.3各项功能

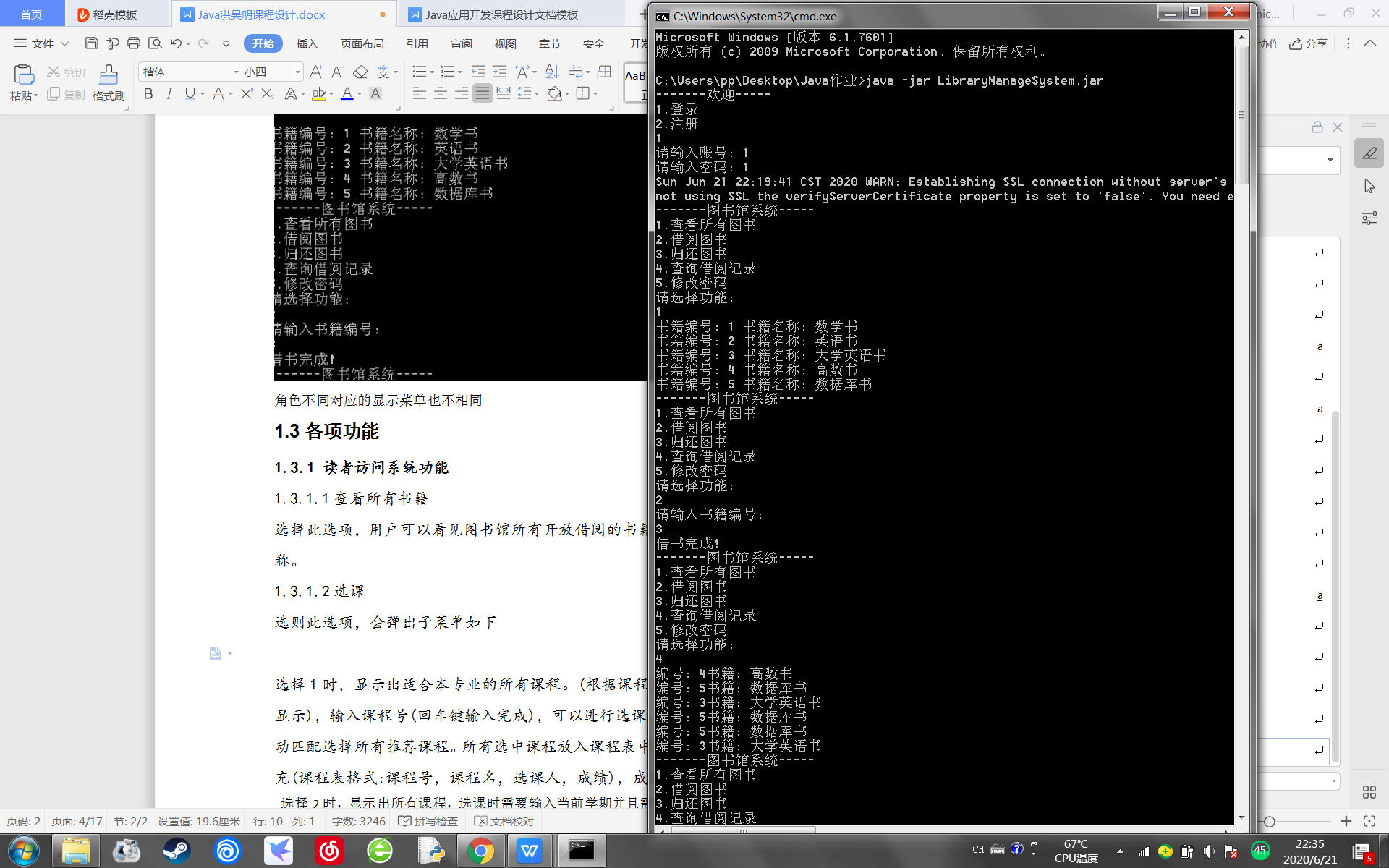
### 1.3.1 读者访问系统功能

1.3.1.1查看所有书籍

选择此选项，用户可以看见图书馆所有开放借阅的书籍。显示书籍编号，书籍名称。

1.3.1.2借书

选则此选项，会弹出子菜单如下



选择书籍的编号（回车键输入完成)，可以进行借阅。

1.3.1.3还书

选择后出现所有已经借阅的书籍，输入后便可归还给图书馆。

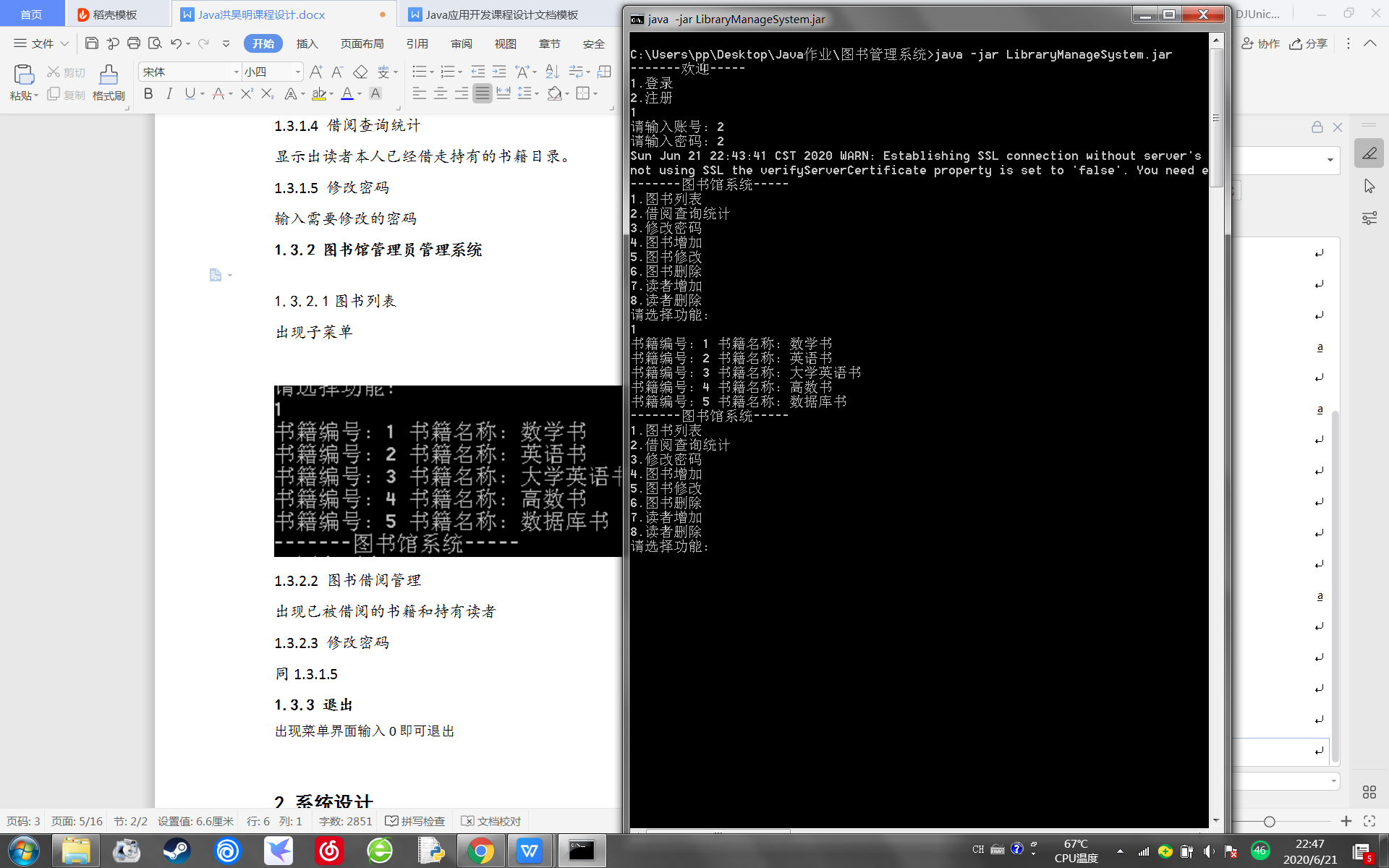
1.3.1.4 借阅查询统计

显示出读者本人已经借走持有的书籍目录。

1.3.1.5 修改密码

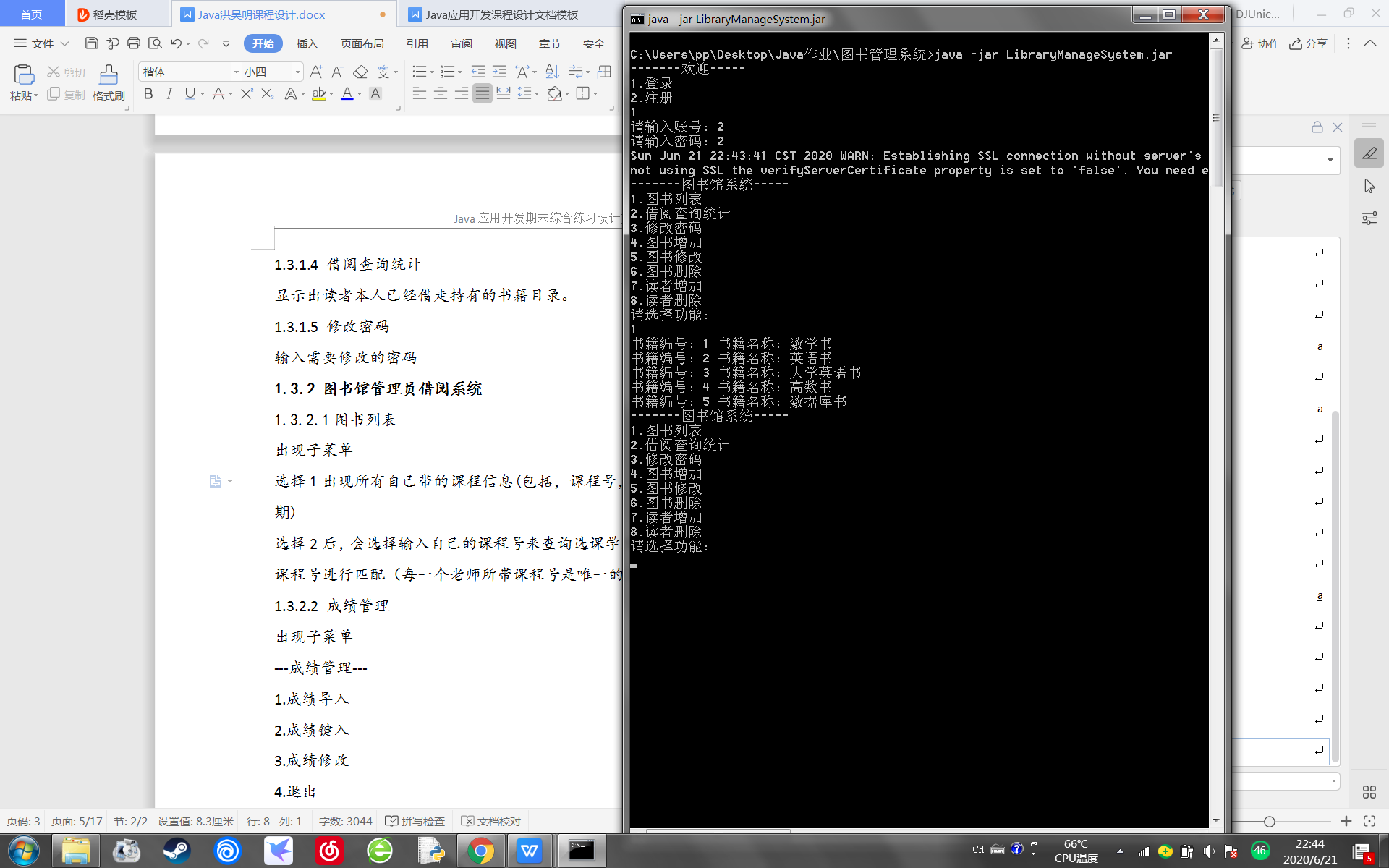
输入需要修改的密码

### 1.3.2 图书馆管理员管理系统



1.3.2.1图书列表

出现子菜单



1.3.2.2 图书借阅管理

出现已被借阅的书籍和持有读者

1.3.2.3 修改密码

同1.3.1.5

### 1.3.3 退出

出现菜单界面输入0即可退出

# 2 系统设计

## 2.1用户用例图

系统分为收银员，管理员两个角色，各角色的用户用例如下图2-1所示：

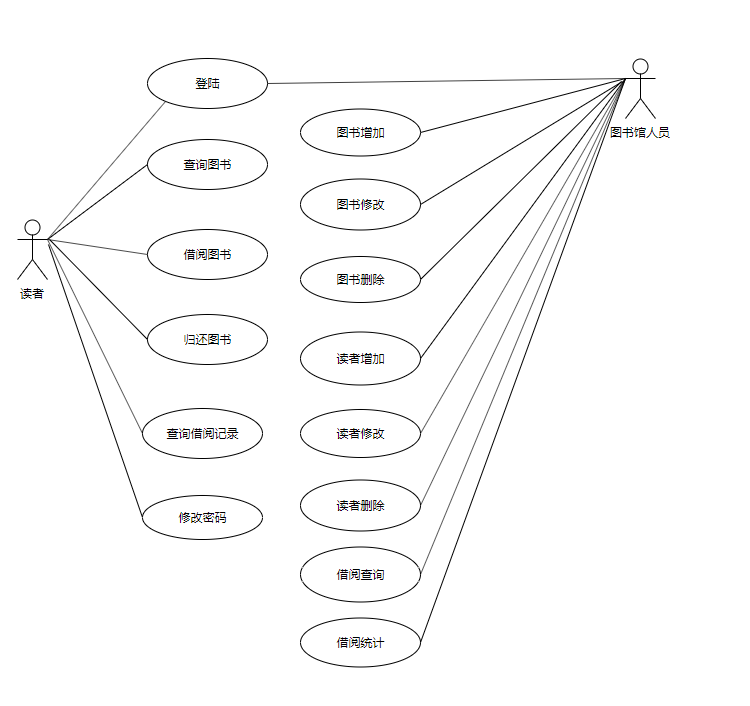


图2-1 用户用例图

## 2.2 UML类图（Class Diagram）

对于用户功能模块，共设计如下6个类。

* 数据库连接类DatabaseConnection：连接数据库并获取连接对象，关闭连接对象。
* VO实体类User：与数据库结构进行映射的类。主要由属性，setter, getter方法组成，VO类中的属性与表中的字段相对应，每一个VO类的对象都表示表中的每一条记录
* DAO接口IUserDAO：主要定义操作的接口，定义一系列数据库的原子性操作，例如增删改查（通常称为CRUD）等。
* DAO实现类UserDAOImpl：DAO接口的真实实现类，主要完成具体数据库操作，但不负责数据库的打开和关闭。
* DAO工厂类DAOFactory：通过工厂类取得一个DAO的实例化对象
* 业务逻辑实现类UserService：对于数据层的原子操作进行整合。还要负责数据库的打开与关闭（不管是否出异常，数据库都要关闭）

各类的结构及类之间的关系如图2-5所示：

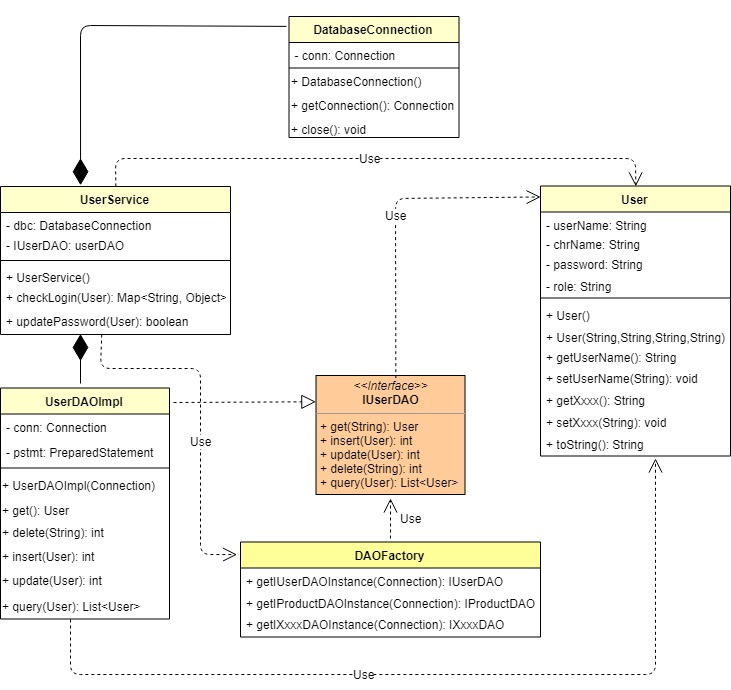
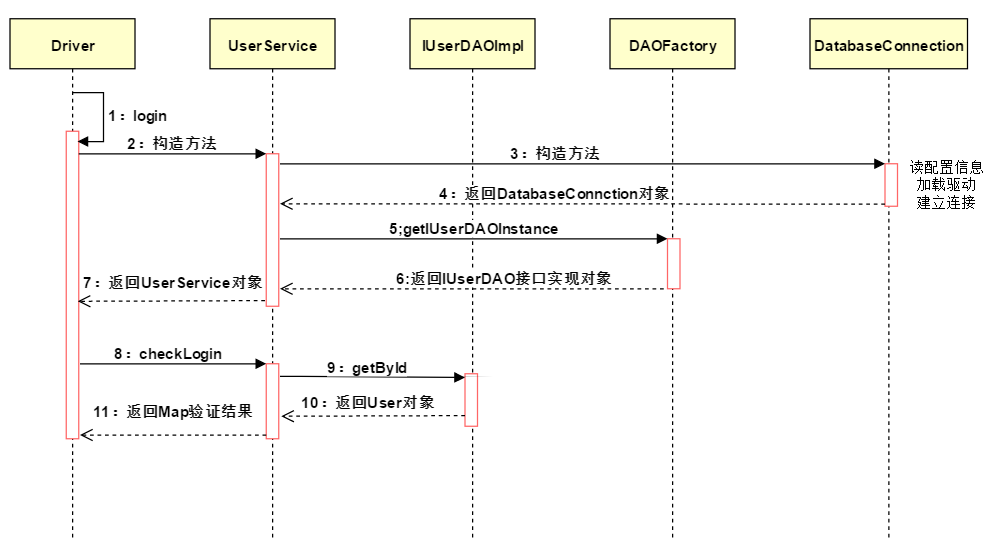


图2-5 用户功能类图

## 2.3UML时序图（Sequence Diagram）2.4.1 系统登录



## 2.5 UML活动图（Activity Diagram）

### 2.5.1 登陆

登录操作需要验证用户名，密码，只有当用户名和密码同时存在即用户存在的情况下才能成功登录系统，否则会提示错误信息，具体流程如图2-6所示。

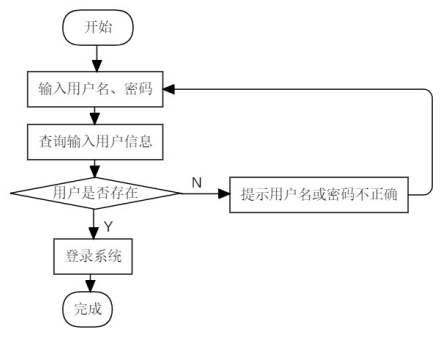


图2-2 登录流程图

## 3. 配置文件

在项目下新建Source Folder，名为resources

### 3.1.1 jdbc.properties文件

该配置文件主要是以文件形式保存数据库的驱动类名称，连接数据库的URL地址，访问数据库的用户名及对应的密码，程序运行时会读取该文件相关信息，避免硬编码，当相关信息发生变化时，只需修改配置文件而不用修改源代码，增加程序的可扩展型。

DBDRIVER=com.mysql.jdbc.Driver

DBURL=jdbc:mysql://localhost:3306/testdb?useUnicode=true&characterEncoding=UTF-8

DBUSER=root

PASSWORD=huawei@123

### 3.1.2 log4j.properties文件

描述该文件作用，并对里面的配置进行解释

log4j.rootLogger=DEBUG, stdout

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n

### 3.1.3 mybatis-config.xml文件

描述该文件作用，并对里面的配置进行解释

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<!-- 根标签 -->

<configuration>

<!-- 指定 资源文件 -->

<properties resource=*"jdbc.properties"*></properties>

<!-- 指定 MyBatis 所用日志的具体实现，未指定时将自动查找-->

<settings>

<setting name=*"logImpl"* value=*"LOG4J"* />

</settings>

<!-- 环境，可以配置多个，default：指定采用哪个环境 -->

<environments default=*"test"*>

<!-- id：唯一标识 -->

<environment id=*"test"*>

<!-- 事务管理器，JDBC类型的事务管理器 -->

<transactionManager type=*"JDBC"* />

<!-- 数据源，池类型的数据源 -->

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*"${DBDRIVER}"* />

<property name=*"url"* value=*"${DBURL}"* />

<property name=*"username"* value=*"${DBUSER}"* />

<property name=*"password"* value=*"${PASSWORD}"* />

</dataSource>

</environment>

</environments>

<!-- 配置mapper映射文件 -->

<mappers>

<mapper resource=*"dao/mapper/UserMapper.xml"* />

</mappers>

</configuration>

## 3.2 VO类User.java

描述该文件作用，并对里面的关键代码进行解释

**public** **class** User {

**private** String userName;

**private** String chrName;

**private** String password;

**private** String role;

//构造方法、getter,setter，toString方法,自行补充

}

## 3.3 DAO接口类IUserDAO.java

描述该文件作用，并对里面的关键代码进行解释

**public** **interface** IUserDAO {

**public** User getById(String userName);

**public** **int** update(User user);

}

## 3.4 接口映射文件UserMapper.xml

描述该文件作用，并对里面的关键信息进行解释

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"dao.IUserDAO"*>

<select id=*"getById"* resultType=*"vo.User"*>

select \* from user where userName = #{userName}

</select>

<update id=*"update"* parameterType=*"vo.User"*>

update user

<set>

<if test=*"chrName != null"*>chrName=#{chrName},</if>

<if test=*"password != null"*>password=#{password},</if>

<if test=*"role != null"*>role=#{role}</if>

</set>

where userName=#{userName}

</update>

</mapper>

## 3.5 工具包Util

### 3.5.1 MybatisUtils.java

描述该文件作用，并对里面的关键代码进行解释

/\*\*

\* 为服务层提供sqlSession对象

\*/

**public** **class** MybatisUtils {

**private** **static** SqlSessionFactory *sqlSessionFactory*;

**public** MybatisUtils() {

**super**();

}

/\*

\* SqlSessionFactory一旦被创建,应该在应用执行期间都存在.在应用运行期间不要重复创建多次

\*/

**static** {

String resourse = "mybatis-config.xml";

**try** {

InputStream inputStream = Resources.*getResourceAsStream*(resourse);

*sqlSessionFactory* = **new** SqlSessionFactoryBuilder().build(inputStream);

} **catch** (IOException e) {

e.printStackTrace();

}

}

**public** **static** SqlSession getSqlSession() {

// 如果不设置参数或者参数为false就是手动提交事务，参数设置为true就是自动提交事务

SqlSession sqlSession = *sqlSessionFactory*.openSession();

**return** sqlSession;

}

}

### 3.5.2 MD5Util.java

该类主要实现对数据库中的用户密码进行MD5加密，描述该文件作用，并对里面的关键代码进行解释

**public** **class** MD5Util {

**public** **static** String md5(String str) {

**byte**[] digest = **null**;

**try** {

MessageDigest md5 = MessageDigest.*getInstance*("md5");

digest = md5.digest(str.getBytes("utf-8"));

} **catch** (NoSuchAlgorithmException e) {

e.printStackTrace();

} **catch** (UnsupportedEncodingException e) {

e.printStackTrace();

}

//16是表示转换为16进制数

String md5Str = **new** BigInteger(1, digest).toString(16);

**return** md5Str;

}

**public** **static** **void** main(String[] args) {

System.***out***.println(*md5*("admin"));

}

}

## 3.6 服务层UserService.java

描述该文件作用，并对里面的关键代码进行解释

**public** **class** UserService {

**private** IUserDAO userDAO = **null**;

**private** SqlSession sqlSession = **null**;

**public** UserService() {

**this**.sqlSession = MybatisUtils.*getSqlSession*();

**this**.userDAO = sqlSession.getMapper(IUserDAO.**class**);

}

// 登录检查，Map形式返回检查结果

**public** Map<String, Object> checkLogin(User user) **throws** Exception {

Map<String, Object> mapResult = **new** HashMap<String, Object>();

**try** {

User foundUser = **this**.userDAO.getById(user.getUserName());

**if** (foundUser == **null**) {

mapResult.put("code", 1);

mapResult.put("msg", "用户名不存在！");

} **else** {

String md5Password = MD5Util.*md5*(user.getPassword());

**if** (!foundUser.getPassword().equals(md5Password)) {

mapResult.put("code", 1);

mapResult.put("msg", "密码不正确！");

} **else** {

mapResult.put("code", 0);

mapResult.put("msg", "登录成功！");

mapResult.put("user", foundUser);

}

}

} **catch** (Exception e) {

mapResult.put("code", 1);

mapResult.put("msg", e.getMessage());

} **finally** { // 无论是否有异常，都需要关闭数据库会话

sqlSession.close();

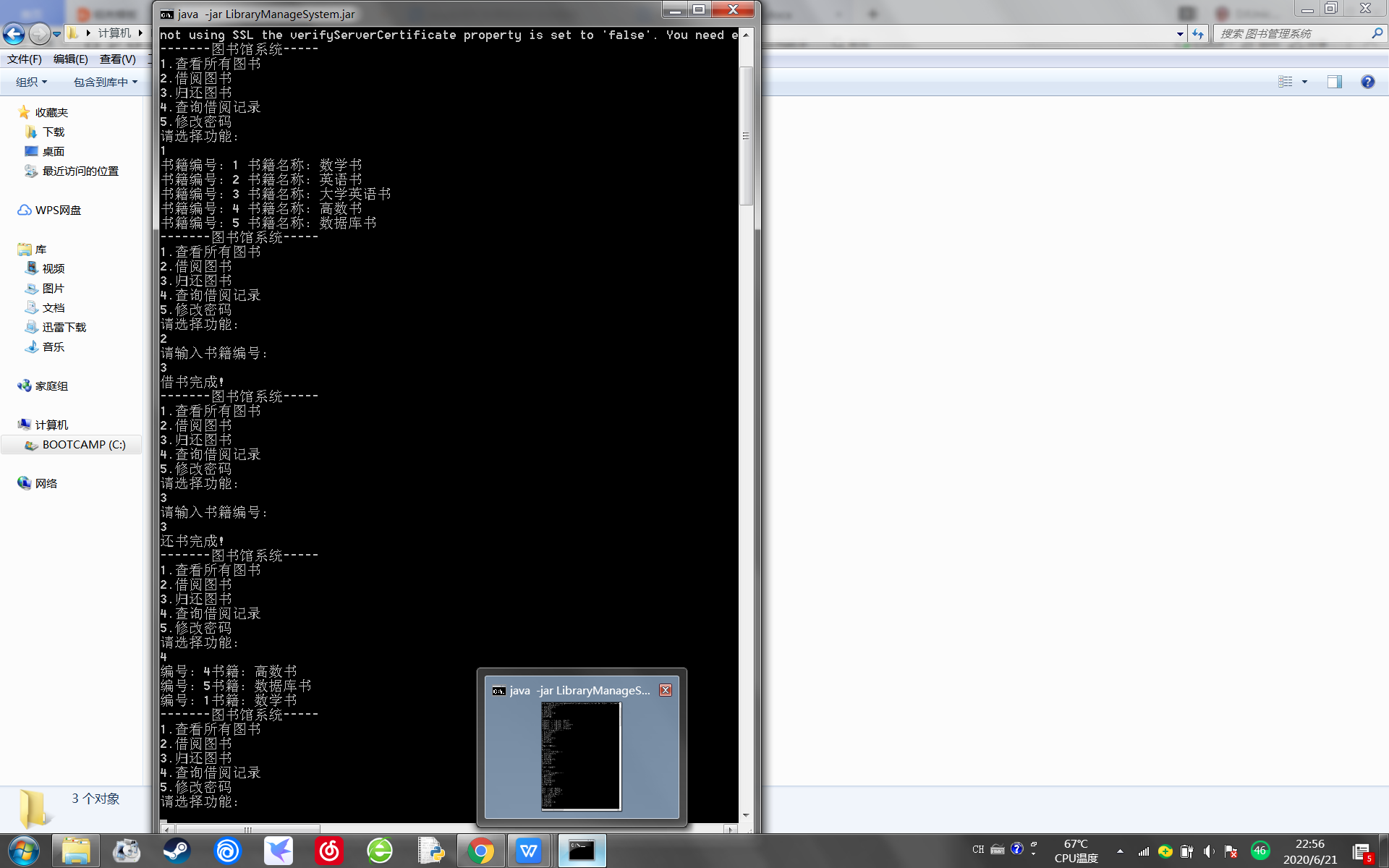
}

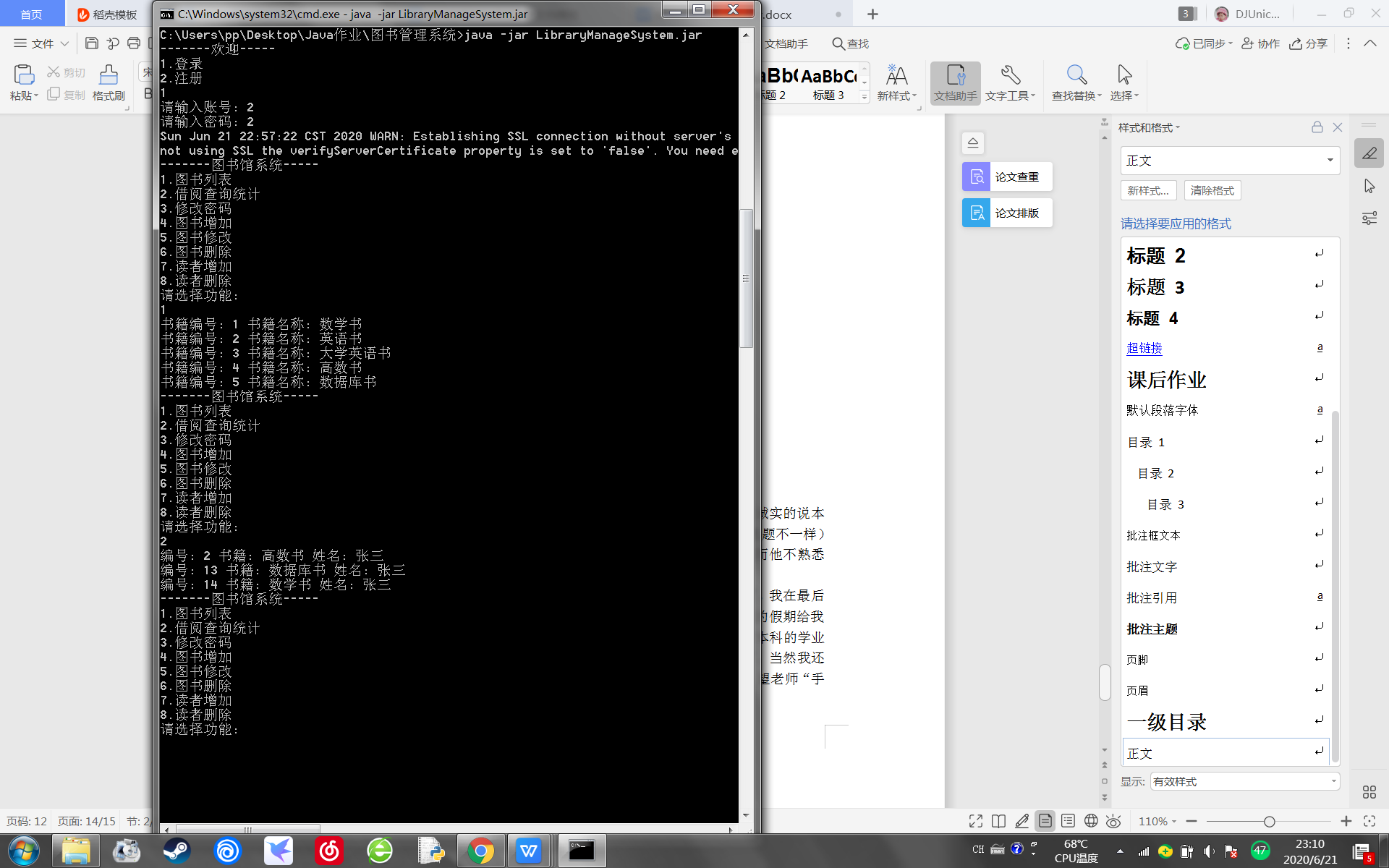
**return** mapResult;

}

}

# 4 系统测试





# 5 系统总结

很多话已经放在前言（目录下方）里说了，这里我也不加赘述了，希望在以后的学习里能够更进一步吧。